Projet d'algorithmique

Sujet : Les aventuriers du Prism ou le problème de sac à dos

1 Présentation

1.1 Enoncé

Le but de ce projet est de spécifier et programmer en langage C un ensemble de procédures qui permettent le traitement informatique du "célèbre" problème du sac à dos. Pour illustrer ce problème, considérons un groupe d'aventuriers du laboratoire Prism (vos enseignants les plus téméraires!) qui participent à un jeu télévisé consistant à se rendre sur une île déserte avec seulement quelques élements en poche et à survivre sur place...Chaque candidat dispose d'un sac à dos dans lequel il peut emporter une certaine quantité d'objets parmi une liste d'objets qui lui est proposée. Chaque objet porte un nom, a un poids, et représente une certaine valeur (par rapport au jeu). Le problème du candidat est de choisir dans cette liste les objets qu'il emportera dans son sac, sachant qu'il cherche à maximiser la valeur des objets à emporter, et veille à ne pas dépasser le poids maximal que peut contenir son sac à dos. Par exemple, voici une liste d'objets avec leur nom, poids et valeur :

nom	poids (en g)	valeur
lampe de poche	68	10
boite d'allumettes	50	7
lunettes de soleil	70	7
livre	150	12
briquet	25	8
couteau	65	5
chapeau	12	3
boussole	18	14
ficelle	5	17
crayon	15	9
bombe anti-moustique	84	6
	$poids_{max}$=195	

1.2 Travail demandé

1. On vous demande d'écrire une ou plusieurs procédures capables de gérer les données du problème. Ces données peuvent être lues dans un fichier ou rentrées par l'utilisateur au clavier. L'utilisateur doit pouvoir modifier l'une ou l'autre des valeurs (poids, valeur) du problème et supprimer/ajouter un (des) objet(s). Pour le projet, on considérera que les poids, et les valeurs des objets sont des entiers.

2. Il existe des configurations où le problème du sac à dos a une solution optimale triviale, quels sont ces cas? Dans la gestion de vos données, vous devez vous assurez que l'un de ces cas n'est pas vérifié avant d'entreprendre la suite.

2 Modélisation

2.1 Principe

Ce problème du sac à dos peut se modéliser simplement sous forme d'un programme linéaire (inutile d'avoir peur! C'est facile à comprendre!). Prenons une configuration avec 4 objets comme ci-dessous :

nom	poids (en g)	valeur	variable
objet1	3	1	x_1
objet2	2	1	x_2
objet3	4	5	x_3
objet4	2	3	x_4
	$poids_{max}$=5		

On associe à chaque objet une variable booléenne x_i qui prend la valeur 1 si l'objet est emporté dans le sac, 0 sinon. L'objectif est de maximiser la somme $x_1 + x_2 + 5x_3 + 3x_4$ sous la contrainte du respect du poids maximal : $3x_1 + 2x_2 + 4x_3 + 2x_4 \leq 5$. De façon plus générale, si on note I l'ensemble des n objets de la liste, a_i et c_i leur poids et valeur respectif, et b le poids maximal, le problème se formule ainsi :

$$Max \sum_{i \in I} c_i x_i$$

$$\text{sous : } \sum_{i \in I} a_i x_i \leq b$$

$$\text{avec : } x_i \in \{0, 1\}, i = 1..n$$

Pour analyser toutes les solutions d'un problème de ce type et fournir celle de meilleure valeur, cela peut prendre plus de 3 ans pour un "simple" problème de 50 variables. Fort heureusement, il existe à l'heure actuelle des méthodes plus efficaces pour trouver la solution optimale ou s'en approcher. Dans ce projet, vous devrez développer ces méthodes de résolution (elles sont décrites dans la suite). Dans l'exemple donné avec 4 objets, la solution optimale est $x* = (0, 0, 1, 0)$ et a pour valeur optimale 5. Pour les méthodes (algorithmes 2, 3 et 4) décrites ci-dessous, vous devez être capable d'afficher la solution et la valeur correspondante. Par exemple,
> Mon sac à dos contient :
> chapeau
> boussole
> lunettes de soleil
> son poids est de : 100
> sa valeur est égale à : 24

3 Algorithme 1 : Recherche rapide d'une borne supérieure de la valeur optimale

3.1 Principe

Pour avoir une idée rapide de la valeur optimale (ou du moins une borne supérieure), il existe un algorithme qui se présente ainsi :

- Trouver un ensemble U d'objets et un objet i vérifiant les propriétes suivantes :
  1. $\forall j \in U, \frac{c_j}{a_j} \geq \frac{c_i}{a_i} \geq \frac{c_l}{a_l} \quad \forall l \in L = I \setminus \{J \cup i\}$
  2. $\sum_{j \in U} a_j \leq b \leq \sum_{j \in U} a_j + a_i$
- Mettre à 1 les variables x_j des objets appartenant à U : $\forall j \in U, \bar{x}_j = 1$
- $\bar{x}_i = \frac{(b - \sum_{j \in U} a_j)}{a_i}$
- $\bar{x}_l = 0 \quad \forall l \in L$
- La valeur de la borne supérieure est donnée par : $\bar{v} = \sum_{j \in U} c_j + c_i \frac{(b - \sum_{j \in U} a_j)}{a_i}$

Pour rechercher l'ensemble U dans l'algorithme, on réalise un tri rapide (type "quicksort") un peu particulier en triant le tableau des rapports ($\frac{c_i}{a_i}$) et en s'arrêtant dès que la somme des poids des p premiers objets de plus grand rapport dépasse le poids maximal du sac (b). Voici une illustration de ce tri sur l'exemple de la partie 2 :

c	1	1	5	3
a	3	2	4	2
c/a	0.33	0.5	1.25	1.5

Après une première étape de tri rapide (en considérant la valeur c/a la plus à droite dans le tableau commme la valeur pivot, et en faisant un tri par ordre décroissant), on obtient :

c	3	1	5	1
a	2	2	4	3
c/a	1.5	0.5	1.25	0.33

La partie du tableau (à gauche du pivot, pivot inclus) contient un seul élement ($c_4 = 3$ et $a_4 = 2$) et le poids total est inférieur à b ($a_4 = 2 \leq 5$). Il faut donc poursuivre le tri. Apres une seconde étape de tri rapide (le pivot est ($c_1 = 1$ et $a_1 = 3$)), on obtient :

c	3	1	5	1
a	2	2	4	3
c/a	1.5	0.5	1.25	0.33

Rien n'a changé, il faut poursuivre le tri sur le sous-tableau composé des 2 éléments ($(c_2 = 1, a_2 = 2)$ et $(c_3 = 5, a_3 = 4)$).

c	3	5	1	1
a	2	4	2	3
c/a	1.5	1.25	0.5	0.33

Cette fois-ci, tout le tableau est trié et on constate qu'en considérant les deux premiers éléments du tableau ($(c_4 = 3, a_4 = 2)$ et $(c_3 = 5, a_3 = 4)$ (éléments à gauche du pivot + pivot), le poids total ($a_4 + a_3 = 2 + 4$) dépasse la valeur de b ($b = 5$). On s'arrête. L'objet correspondant au premier élement du tableau appartient à U. L'objet correspondant à l'élement suivant correspond à l'objet $\{i\}$ de l'algorithme (objet "frontière"), les autres objets appartiennent à l'ensemble L. On en déduit :$\bar{x}_4 = 1$, $\bar{x}_3 = \frac{(5-2)}{4} = \frac{3}{4}$, $\bar{x}_1 = \bar{x}_2 = 0$ et $\bar{v} = 5 * \frac{3}{4} + 3 * 1 = \frac{27}{4} = 6.75$.

Prenons un autre exemple avec 8 objets et un poids maximal égal à 43. Pour cet exemple, une seule étape de l'algorithme de tri rapide suffit à déterminer l'ensemble U, l'objet "frontière" i et l'ensemble L.

c	10	13	5	14	6	13	8	10
a	7	10	5	7	13	7	6	8
c/a	1.42	1.3	1	2	0.46	1.85	1.33	1.25

Apres une première étape de tri (avec le dernier élément comme pivot), on obtient :

c	10	13	8	14	13	10	5	6
a	7	10	6	7	7	8	5	13
c/a	1.42	1.3	1.33	2	1.85	1.25	1	0.46

Si on additionne le poids des éléments à gauche du pivot sans le pivot, on obtient un poids de 37 et si on ajoute le poids du pivot ($c = 10, a = 8$), le poids total (45) dépasse le poids maximal (43). On s'arrête. Les objets de U correspondent aux élements à gauche du pivot. L'objet i correspond à l'élement pivot et le reste des objets appartiennent à L.

3.2 Travail demandé

1. Vous devez expliquer pourquoi cet algorithme fournit la borne supérieure de la valeur optimale du problème.
2. Vous devez écrire une ou plusieurs procédures capables de réaliser le tri illustré précédemment, et de fournir, d'après l'algorithme ci-dessus, la valeur de la borne supérieure et la solution continue correspondante (valeur des $\bar{x}_i$, $\forall i \in I$).

4 Algorithme 2 : Recherche d'une solution approchée par la méthode gloutonne

4.1 Principe

Pour trouver une solution au problème de sac à dos qui soit "proche" de la solution optimale, on utilise l'algorithme dit glouton suivant (en reprenant les résultats de l'algorithme donné en 3.1):

- Mettre à 1 les variables x_j des objets appartenant à U : $\forall j \in U, \tilde{x}_j = 1$
- Chercher si on peut continuer à remplir le sac à dos avec des objets de L

4.2 Travail demandé

1. Vous devez écrire une procédure capable de fournir une solution approchée (valeur des $\tilde{x}_i$, $\forall i \in I$) et la valeur correspondante ($\tilde{v} = \sum_{i \in I} c_i \tilde{x}_i$) avec l'algorithme décrit ci-dessus. Expliquer quelle stratégie vous avez utilisé pour continuer à remplir le sac à dos (étape 2 de l'algorithme).
2. Montrer que cette méthode ne donne pas toujours la solution optimale mais fournit une borne inférieure de la valeur optimale (Avec un exemple).

5 Algorithme 3 : Calcul de la solution optimale exacte par programmation dynamique

5.1 Principe

Vous ne devez pas être spécialiste de la programmation dynamique pour comprendre le fonctionnement de cette méthode appliquée au problème du sac à dos. Notons : $V(k, y)$ la meilleure valeur qu'on peut obtenir pour le problème du sac à dos réduit aux k premiers objets et un poids maximal égal à y. Il existe deux types de solutions possibles pour ce problème réduit : celle qui inclue l'objet k, celle qui n'inclue pas l'objet$k$. Pour la solution qui n'inclue pas l'objet k, la valeur optimale obtenue est la même que celle obtenue pour le problème du sac à dos réduit aux (k-1) premiers objets avec un poids maximal de y : $V(k-1, y)$. Pour la solution qui inclue l'objet k, la valeur optimale est égale à la valeur de l'objet k ajoutée à la valeur optimale obtenue pour le problème du sac à dos réduit aux $k-1$ premiers objets et un poids maximal égal à $y - a_k$ $(y - a_k \geq 0)$: $c_k + V(k-1, y-a_k)$. Parmi ces deux solutions possibles, on retiendra celle qui fourni la meilleure valeur. Nous obtenons donc la récurrence suivante :

$$V(k,y) = \begin{cases} \max\{V(k-1,y), c_k + V(k-1, y-a_k)\} & \text{si } y - a_k \geq 0 \\ V(k-1,y) & \text{si } y - a_k < 0 \end{cases}$$

Nous avons également les conditions initiales suivantes :

$$\begin{aligned} V(0,y) &= 0 \quad \text{pour } y \geq 0 \\ V(k,0) &= 0 \quad \text{pour } k \geq 0 \end{aligned}$$

Pour trouver la valeur optimale $V(n,b)$ de la solution optimale du problème initial, il suffit de construire le tableau $V(k,y)$ (pour $k = 0..n$ et $y = 0..b$), à l'aide de la récurrence précédente. Par exemple pour le problème de la partie 2, on a le tableau V suivant :

k|y	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	1	1	1
2	0	0	1	1	1	2
3	0	0	1	1	5	5
4	0	0	3	3	5	5

5.2 Travail demandé

1. Vous devez écrire une procédure permettant de trouver la valeur optimale du problème de sac à dos à l'aide de la programmation dynamique.
2. Quelle est la complexité de cet algorithme?
3. Vous devez écrire une procédure capable de retrouver la solution optimale à partir du tableau V.

6 Algorithme 4 : Calcul de la solution optimale exacte par la technique des couples dominants

6.1 Principe

Le principe général de cet algorithme consiste à générer récursivement des listes L_k de couples (w, p), pour $k = 1, .., n$; où w est un poids et p une valeur. Initiallement, on a : $L_0 = (0, 0)$. Notons N_k l'ensemble des couples générés à chaque étape k, où l'étape k correspond à la prise en compte du $k$ème objet de la liste.

$$N_k = \{(w + w_k, p + p_k) | (w, p) \in L_{k-1}, w + w_k \leq b\}$$

Par exemple, pour le cas simple donné dans la partie 2, à l'étape 0 : $L_0 = \{(0, 0)\}$. Puis à l'étape 1, on s'intéresse à l'insertion de l'objet1 dans le sac. Si on l'insère, on obtient le nouveau couple $(3, 1)$. On a $L_1 = \{(0, 0), (3, 1)\}$. A l'étape 2, on s'intéresse à l'insertion de l'objet2 dans le sac. Si on insère l'objet2 et pas l'objet1, on obtient le nouveau couple $(2, 1)$ généré à partir du couple $(0, 0)$. Si on choisit d'insérer à la fois l'objet1 et l'objet2, on obtient le couple $(5, 2)$ généré à partir du couple $(3, 1)$. On aurait donc par ce principe : $L_2 = \{(0, 0), (3, 1), (2, 1), (5, 2)\}$. Mais il est inutile de garder le couple $(3, 1)$ car il est moins "bon" du point de vue de l'optimalité, du couple $(2, 1)$, dans la mesure où il correspond à une solution plus gourmande en poids ($3 > 2$) pour une même valeur (1). Dans ce cas, on dit que le couple $(3, 1)$ est dominé par le couple $(2, 1)$. On définit plus formellement l'ensemble des couples dominés à l'étape k par :

$$D_k = \{(w, p) | (w, p) \in L_{k-1} \cup N_k, \exists (w', p') \in L_{k-1} \cup N_k \text{ avec } w' \leq w, p \leq p', (w', p') \neq (w, p)\}$$

Par conséquence, on définit récursivement la liste L_k par :

$$L_k = L_{k-1} \cup N_k - D_k$$

Si, à l'interieur d'une liste L_k, on ordonne les couples dans l'ordre croissant de leur poids et valeur, la valeur du couple "le plus grand" (le plus à droite) de la liste L_n correspond à la valeur optimale du problème du sac à dos.
Exemple pour le problème de la partie 2:
$L_0 = \{(0, 0)\}$

$L_1 = \{(0, 0), (3, 1)\}$
$N_2 = \{(2, 1), (5, 2)\}$
$D_2 = \{(3, 1)\}$

$L_2 = \{(0, 0), (2, 1), (5, 2)\}$
$N_3 = \{(4, 5)\}$
$D_3 = \{(5, 2)\}$

$L_3 = \{(0, 0), (2, 1), (4, 5)\}$
$N_4 = \{(2, 3), (4, 4)\}$
$D_4 = \{(2, 1)\}$

$L_4 = \{(0, 0), (2, 3), (4, 5)\}$

6.2 Travail demandé

Vous devez écrire une procédure permettant de trouver la solution et la valeur optimale du problème de sac à dos à l'aide de cette technique des couples dominants. Vous devrez bien détailler la manière dont vous procédez, quelle(s) structure(s) de données vous utilisez et comment vous avez adapté cette technique pour générer la solution optimale.

7 Algorithme 5 : Heuristique pour trouver une solution approchée pour un problème de sac à dos à double contrainte

7.1 Principe

Cette partie de travail est optionnelle. On suppose qu'en plus de la contrainte de poids, on ajoute au problème une contrainte supplémentaire (contrainte de volume par exemple). Celle-ci peut s'écrire par exemple :

$$\sum_{i \in I} v_i x_i \leq v$$

7.2 Travail demandé

Proposer et programmer une méthode approchée permettant de trouver une solution réalisable (qui vérifie les 2 contraintes, de poids et de volume) à ce nouveau problème et qui soit proche de l'optimum (de la valeur optimale). Tester cette nouvelle méthode sur plusieurs jeux de données.

8 Obligations et recommandations

- Ce projet est à réaliser en binôme. (Les 2 personnes doivent appartenir au même groupe de TD et suivre le même TD, sauf exception à notifier explicitement auprès de son chargé de TD et du responsable de cours)
- Vous devrez rédiger un rapport de 5 à 10 pages qui présentera votre réalisation.
- Vous préciserez les structures de données que vous avez utilisées en les justifiant si nécessaire.
- Vous présenterez chaque procédure en indiquant le mieux possible les raisons de votre choix pour telle procédure (ou fonction).

Du point de vue programmation, vous devrez :

- documenter chaque procédure (fonction) dans le code
- fournir un(des) jeu(x) de tests (cas nominaux ou erreurs) validant chaque fonctionnalité

Vous êtes libres d'enrichir à votre guise le projet à condition que le travail demandé (les 4 premiers algorithmes) soit déjà traité.
Vous pouvez poser des questions ou aborder certains points difficiles avec votre chargé de TD, et exceptionnellement avec votre responsable de cours.
Ce projet (rapport) est à rendre pour la semaine du : 22 Mai - 26 Mai 2006 à votre chargé de TD.
Vous présenterez votre travail lors d'une soutenance de 10 minutes (5 minutes de présentation + 5 minutes de question) en séance de TD.